Colegiul Național de Informatică “Tudor Vianu”

LUCRARE DE ATESTAT

INFORMATICĂ

TEMA:

RIDE - MEDIU DE DEZVOLTARE RUST

COORDONATOR:
Prof. POPESCU CRISTIANA

CANDIDAT:
BURTESCU EDUARD-MIHAI

TEMA LUCRĂRII:

RIDE - MEDIU DE DEZVOLTARE RUST

CUPRINS

INTRODUCERE

Ce este Rust?

Rust[1] este un limbaj nou de programare pentru sisteme, o nișă ocupată până recent de C și C++, dar și pentru aplicații sau servicii, oferind performață și predictabilitate unde Java, C# sau alte limbaje de nivel mai înalt ar fi folosite în mod tradițional.

Descrierea oficială:

> Rust is a systems programming language that runs blazingly fast, prevents almost all crashes, and eliminates data races.
>
> – http://rust-lang.org [1]

Exemplu de cod:

```rust
// This code is editable and runnable!
fn main() {
    // A simple integer calculator:
    // `+` or `-` means add or subtract by 1
    // `*` or `/` means multiply or divide by 2

    let program = "+ + * - /";
    let mut accumulator = 0;

    for token in program.chars() {
        match token {
            '+' => accumulator += 1,
            '-' => accumulator -= 1,
            '*' => accumulator *= 2,
            '/' => accumulator /= 2,
            _ => { /* ignore everything else */ }
        }
    }

    println!("The program \"{}\" calculates the value {}",
             program, accumulator);
}
```

Prima versiune stabilă (1.0) va fi lansată pe 15 mai 2015, după 6 ani de dezvoltare sponsorizată de Mozilla, cu peste 1000 de contributori de-a lungul acestora.

De ce Rust?

Rust poate sa înlocuiască C şi C++ în multe domenii unde un nivel de control sau performanţă este necesar, dar siguranţa oferită chiar şi de ultimele versiuni C++ lasă de dorit.

De exemplu, toate browserele web sunt scrise, în cea mai mare parte, în C++ (cu unele componente în C). Rezultatul?

> În 2014, 1035 de vulnerabilităţi au fost descoperite în primele 5 dintre cele mai populare browsere: Google Chrome, Mozilla Firefox, Internet Explorer, Opera şi Safari. Acest număr reprezintă o creştere de 42% faţă de 2013.
>
> – Secunia Vulnerability Review 2015 [4]

O mare parte dintre aceste vulnerabilităţi au la bază greşeli în administrarea şi accesarea memoriei, ce pot fi prevenite de Rust la momentul compilării (*memory safety*), chiar şi atunci când sunt folosite multiple fire de execuţie ce partajează memoria, în mod paralel sau concurent.

În această direcţie, Mozilla şi Samsung lucrează la un nou prototip de browser, numit Servo[3], folosind Rust pentru a exploata paralelismul în randarea paginilor web şi pentru a reduce semnificativ suprafaţa de atac. Un raport[5] din martie 2015 indică o creştere a performanţei de 2-4 ori mai mare faţă de Firefox. De asemenea, consumul de energie este redus, ducând la îmbunătăţirea duratei de utilizare a bateriei pentru dispozitivele mobile.

De ce un IDE?

Deşi proiecte impresionant de mari (Servo şi compilatorul Rust au fiecare în jur de 400 de mii de linii de cod Rust) au fost scrise fără ajutorul unor unelte specializate, lipsa unui mediu integrat de dezvoltare (IDE) dedicat limitează progresul şi îngreunează familiarizarea începătorilor cu limbajul.

Până la această dată nu există un IDE complet pentru Rust, doar câteva proiecte independente, ce nu refolosesc compilatorul Rust pentru analiza sintactică şi semantică. Acest fapt rezultă în discrepanţe între implementarea completă a limbajului (reprezentată de compilator) şi acele IDE, limitând capacitatea lor de a înţelege un program oarecare scris în Rust, şi de a asista programatorul.

Consider că folosirea compilatorului Rust într-un IDE este viabilă şi că acesta poate pune la dispoziţie toată întelegerea semantică folosită în mod normal pentru compilarea unui program, fară a necesita reimplementarea oricărei părţi semnificative pentru a atinge acest scop.

Am decis astfel să dezvolt o aplicaţie ce demonstrează utilizarea compilatorului Rust într-un IDE. Numele de “ridɛ” este un acronim pentru “Rust IDE”, deşi aplicaţia este un prototip, nu un IDE complet.

PROGRAMELE UTILIZATE

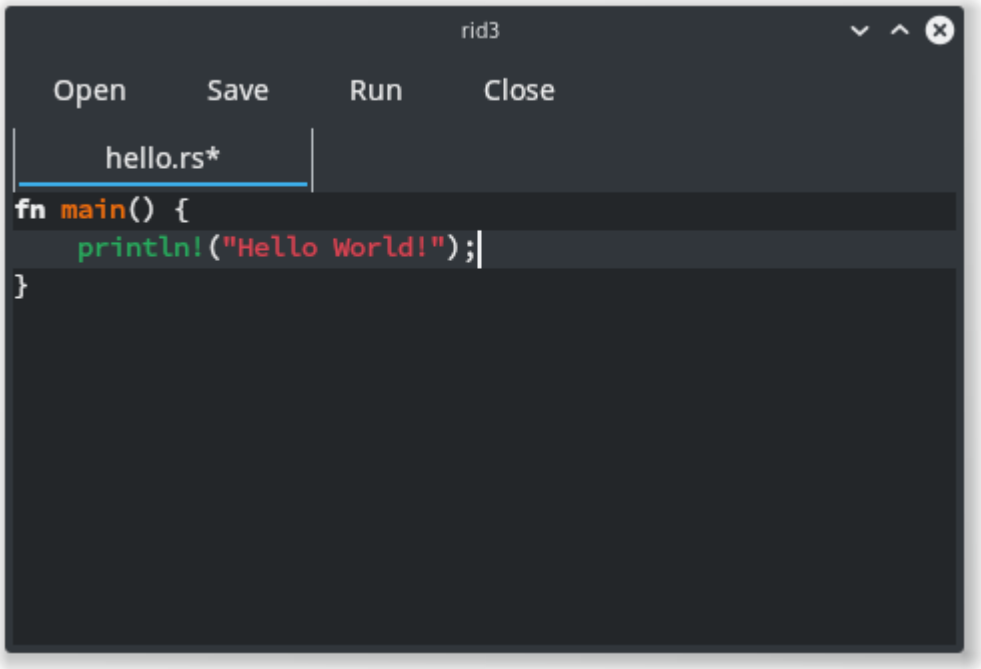
Pentru scrierea codului, am folosit Kate, un editor fară facilităţi specifice unui limbaj anume, dar cu o selecţie bogată de limbaje pentru care colorarea sintactică este suportată, inclusiv Rust. Abilitatea de a căuta şi înlocui în mai multe fişiere în acelaşi timp, folosind expresii regulate, este foarte utilă pentru proiecte mai mari. De asemenea, consola şi navigatorul de fişiere integrate facilitează dezvoltarea oricărui proiect.

Navigarea resurselor de pe Internet, în special documentaţia Rust[2], a fost efectuată în Chromium, versiunea open-source a browserului Google Chrome.

Versiunile exacte ale compilatorul Rust folosit, rustc 1.1.0-nightly (f9e53c7f2 2015-04-24) şi ale administratorului de dependenţe, cargo 0.2.0-nightly (dac600c 2015-04-22), se pot găsi la adresa http://static.rust-lang.org/dist/2015-04-25/.

Am compilat şi testat executabilului de Windows într-o maşină virtuală, folosind VirtualBox, iar acest document a fost scris în LibreOffice Writer.

PREZENTAREA APLICAȚIEI



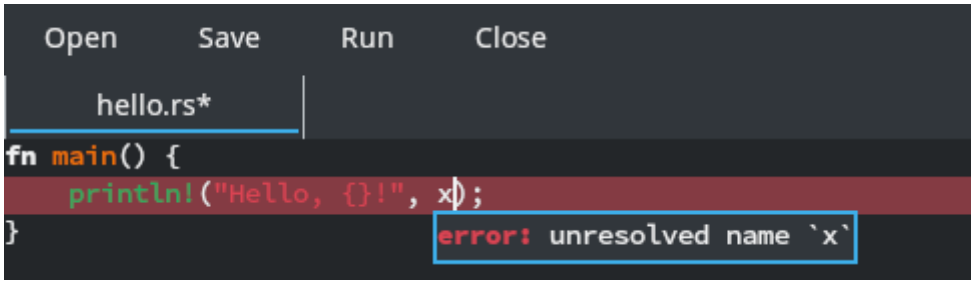
Interfața aplicației este constituită din 3 elemente principale:

1. Bara de acțiuni, cu 4 butoane:
    1. **Open**: deschide un fișier într-un tab nou.
    2. **Save**: salvează fișierul din tabul curent.
    3. **Run**: salvează, compilează și rulează fișierul curent.
    4. **Close**: închide tabul curent, dacă fișierul nu are modificări.
2. Bara de taburi: fiecare tab afișează numele fișierului, urmat de un asterisc dacă fișierul are modificări.
3. Editorul de cod: se pot observa colorarea sintactică, cursorul de text și evidențierea liniei de sub acesta.

Diacritice și caractere speciale Unicode sunt suportate:

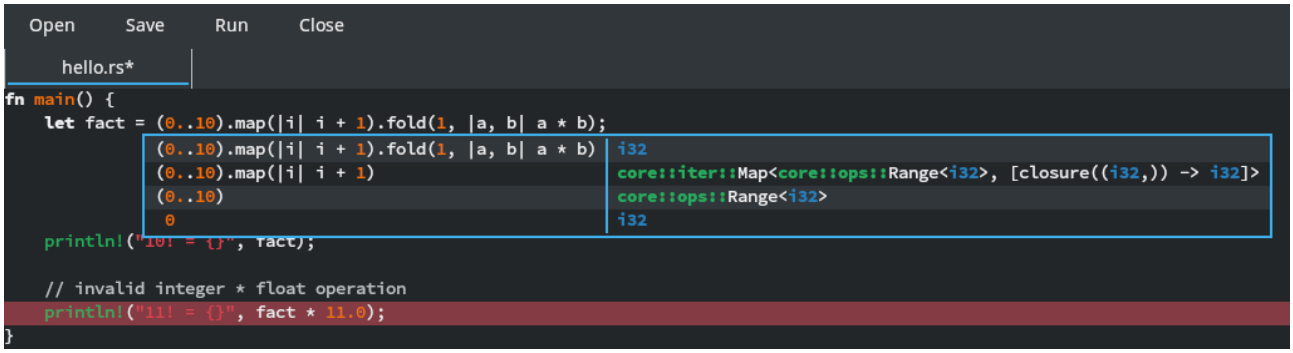
```
fn main() {
    println!("Bună dimineața! Afară sunt {}°C", 15.0);
}
```

La deschiderea unui fişier, şi dupa modificări, conţinutul acestuia este trecut prin compilator. Dacă există o eroare de compilare, linia respectivă va fi evidenţiată cu un fundal roşu, iar mesajul de eroare va fi afişat la plasarea cursorului deasupra liniei.



În abesenţa erorilor, tipul oricărei expresii poate fi obţinut prin acelaşi mecanism de plasare a cursorului asupra locaţiei dorite.

De asemenea, tipurile expresiilor sunt disponibile chiar daca o eroare a intervenit ulterior, în afara linilor afectate de eroare:



Compilatorul rulează în paralel, fluiditatea interacţiunilor cu editorul prin intermediul tastaturii şi mousului nefiind afectate de execuţia acestuia în fundal.

IMPLEMENTARE

Aplicația are în total peste 4000 de linii de cod, cea mai mare parte fiind interfața grafică, cu un întreg algoritm de rezolvare a sistemelor de inecuații lineare ce descriu aranjamentul elementelor în fereastră – un sistem mult mai flexibil decât s-a dovedit necesar.

În dreapta se află procedura de inițializarea a compilatorului, cu toate fazele ei aferente.

```
let krate = driver::phase_1_parse_input(&sess, cfg, &input);

still_alive!();
let krate = driver::phase_2_configure_and_expand(&sess, krate, "r3", None)
    .expect("phase_2_configure_and_expand aborted");

let mut forest = ast_map::Forest::new(krate);
let arenas = ty::CtxtArenas::new();

still_alive!();
let ast_map = driver::assign_node_ids_and_map(&sess, &mut forest);
let krate = ast_map.krate();

still_alive!();
CrateReader::new(&sess).read_crates(krate);
let lang_items = middle::lang_items::collect_language_items(krate, &sess);

still_alive!();
let resolve::CrateMap {
    def_map,
    freevars,
    trait_map,
    ..
} = resolve::resolve_crate(&sess,
                           &ast_map,
                           &lang_items,
                           krate,
                           resolve::MakeGlobMap::No);

// Discard MTWT tables that aren't required past resolution.
syntax::ext::mtwt::clear_tables();

still_alive!();
let named_region_map = middle::resolve_lifetime::krate(&sess, krate, &def_map);

still_alive!();
let region_map = middle::region::resolve_crate(&sess, krate);

//middle::check_loop::check_crate(&sess, krate);

//middle::check_static_recursion::check_crate(&sess, krate, &def_map, &ast_map);

still_alive!();
let tcx = &ty::mk_ctxt(sess,
                       &arenas,
                       def_map,
                       named_region_map,
                       ast_map,
                       freevars,
                       region_map,
                       lang_items,
                       stability::Index::new(krate));
typeck::check_crate(tcx, trait_map);

let _ = tx.send(Res::Done);
```

CONCLUZII

Am arătat posibilitatea utilizării compilatorului Rust într-un IDE, prin extragerea rezultatelor analizei semantice, mai exact tipurile unor expresii cu o complexitate arbitrară, şi prezentarea lor utilizatorului.

În acelaşi timp, am dezvoltat un set de primitive pentru construirea unei interfeţe grafice în Rust, flexibile şi compozabile.

Proiectul va fi dezvoltat în continuare şi poate fi găsit la adresa https://github.com/eddyb/r3.

BIBLIOGRAFIE

[1] Rust – http://rust-lang.org

[2] Documentația Rust – http://doc.rust-lang.org

[3] Servo – https://github.com/servo/servo

[4] Secunia Vulnerability Review 2015 – Browser security – https://secunia.com/resources/vulnerability-review/browser-security/

[5] Experience Report: Developing the Servo Web Browser Engine using Rust – https://kmcallister.github.io/papers/2015-servo-experience-report-draft1.pdf

[6] Wikipedia – http://en.wikipedia.org